ns-3 Training

ns-3 Annual Meeting
June 2017

ns-3 training goals

- Make attendees more productive with ns-3
  - Learn about the project scope, and where to get additional help
  - Understand the architecture and design goals of the software
  - Introduce how to write new code for the simulator
  - Learn about selected topics in more detail
  - Answer your questions

Agenda and Instructors

- Software and usage overview (T. Henderson)
- How to write new models (T. Pecorella)
- Wi-Fi and wireless models (T. Henderson)
- TCP and AQM models (M. Tahiliani)
- Traffic control (S. Avallone)

Please ask questions along the way!

Additional training archives

- LTE (Lorenza Giupponi and Biljana Bojovic), June 2016
- Parallel, Distributed Simulations (Peter Barnes), June 2016
- Direct Code Execution (Tom Henderson), June 2016
- Tracing (Walid Younes), June 2014

http://www.nsnam.org/wiki/Training2017

Your feedback on requested topics

1) what is your past level of experience with ns-3?

- various (from starting the tutorial to having written new models)

2) what technical topics in the simulator interest you the most?

- Wi-Fi, LTE, TCP
- routing 6LoWPAN, IoT, IPv6, BGP, and the core

3) past level of experience with any other network simulation tools?

- MATLAB/Simulink, plus ns-2, OPNET, OMNeT++, Totem

Your feedback on requested topics

4) what do you most want to get out of the training sessions

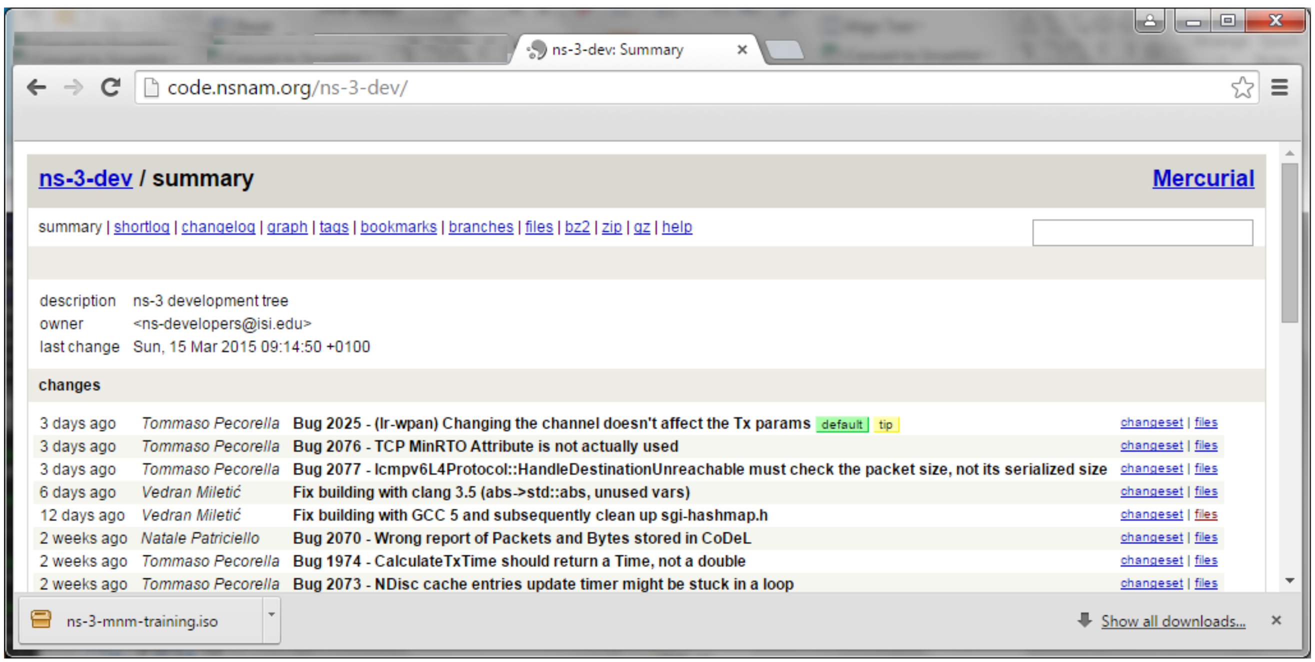
- refresh, get ideas for lab assignments, understand real-time simulations, inject real traffic, global tips and tricks about ns-3, learn LTE, implement new models

Options for working along

1) Download the required packages onto your (Linux, OS X, or BSD) system

2) Download the ISO image (Live DVD)

3) Browse the code online: https://code.nsnam.org



Project overview

Motivations for ns-3 project

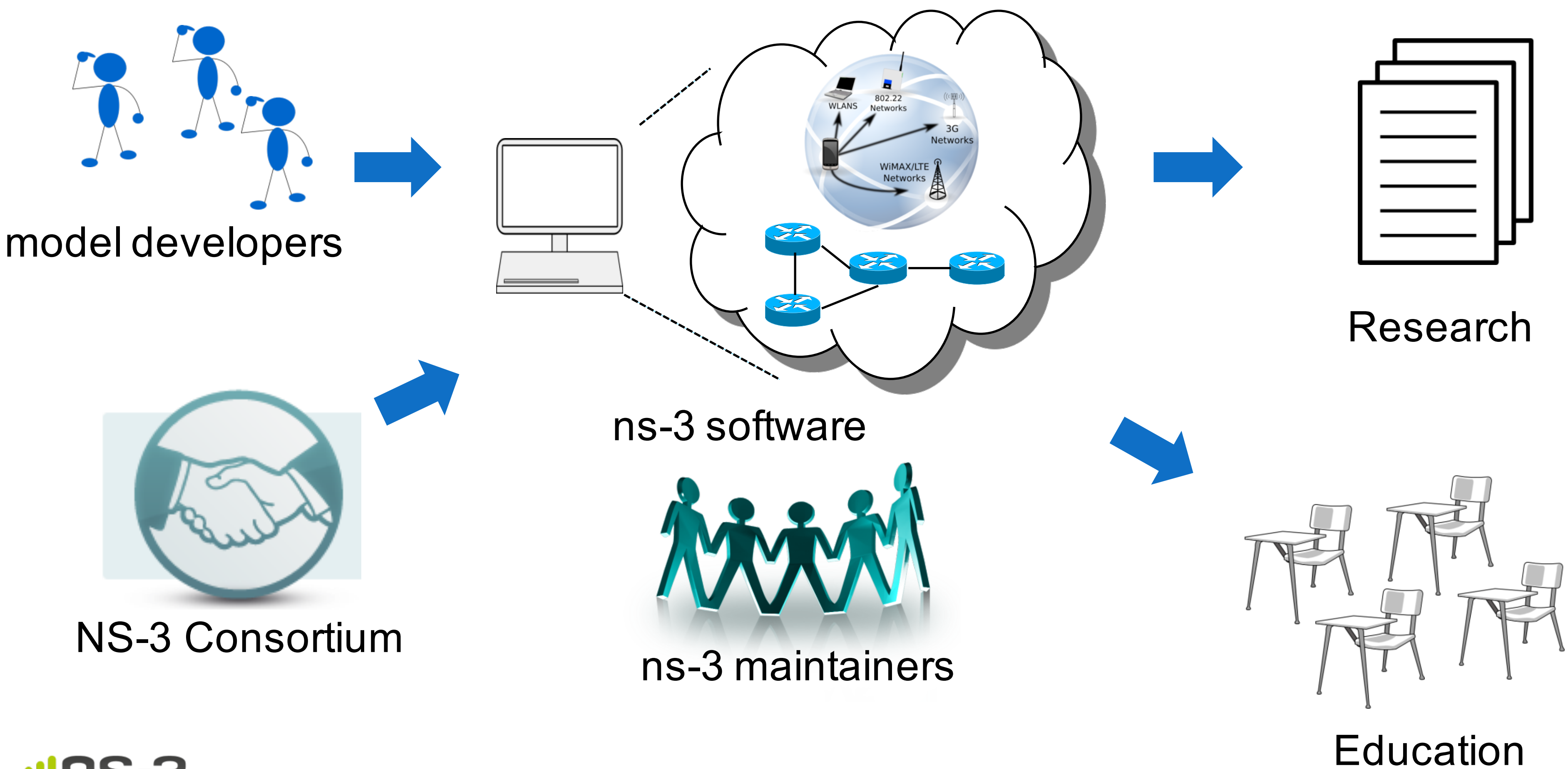
Develop an extensible simulation environment for networking research

1) a tool **aligned with the experimentation needs** of modern networking research
2) a tool that **elevates the technical rigor** of network simulation practice
3) an **open-source project** that encourages community contribution, peer review, and long-term maintenance and validation of the software

Community-maintained, scientific computing software by following best current practices for open source

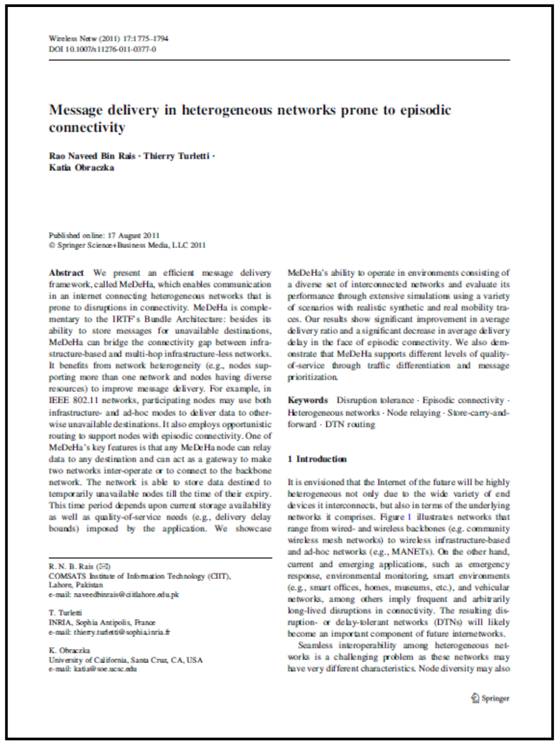
ns-3: An Open Source Network Simulator

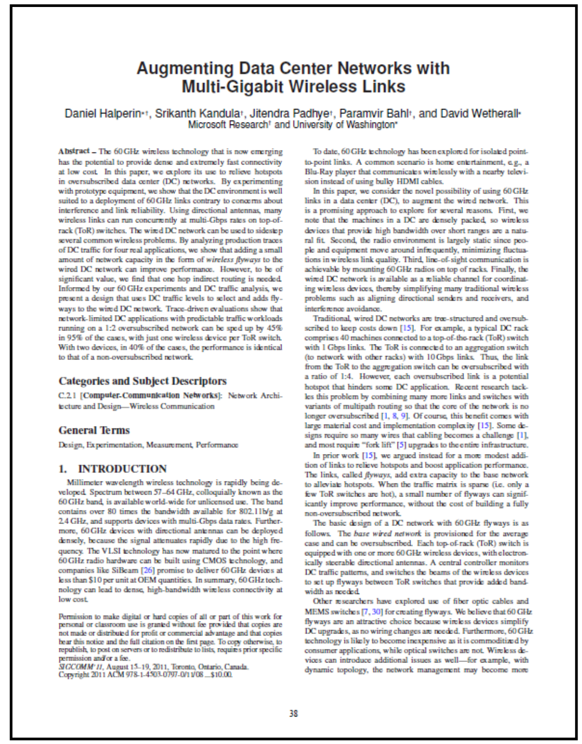
- ns-3 is a ***discrete-event network simulator*** targeted for ***research and educational use***



What have people done with ns-3?

- thousands of publications to date
  - search of 'ns-3 simulator' on IEEE and ACM digital libraries, or Google Scholar

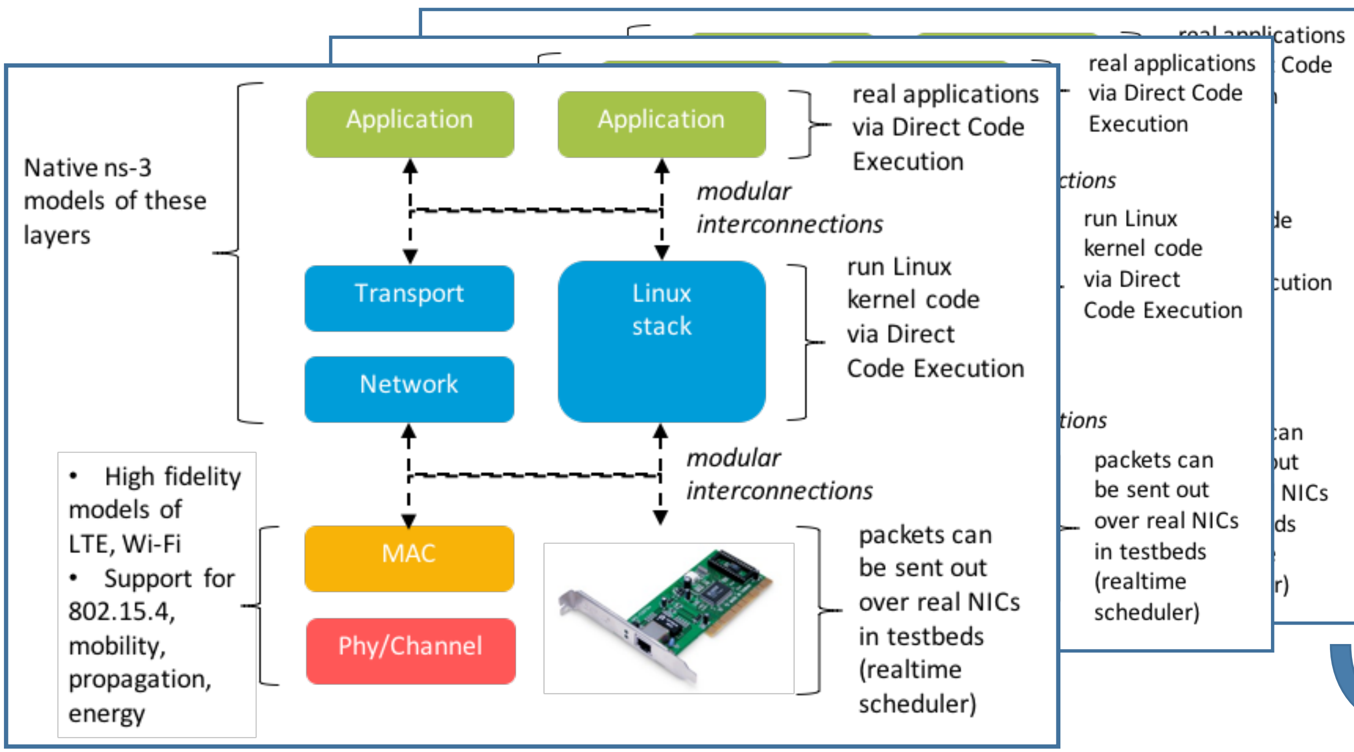




ns-3 overview

- ns-3 is a leading open source, **packet-level network simulator** oriented towards network research, featuring a **high-performance core** enabling **parallelization across a cluster** (for large scenarios), **ability to run real code**, and **interaction with testbeds**

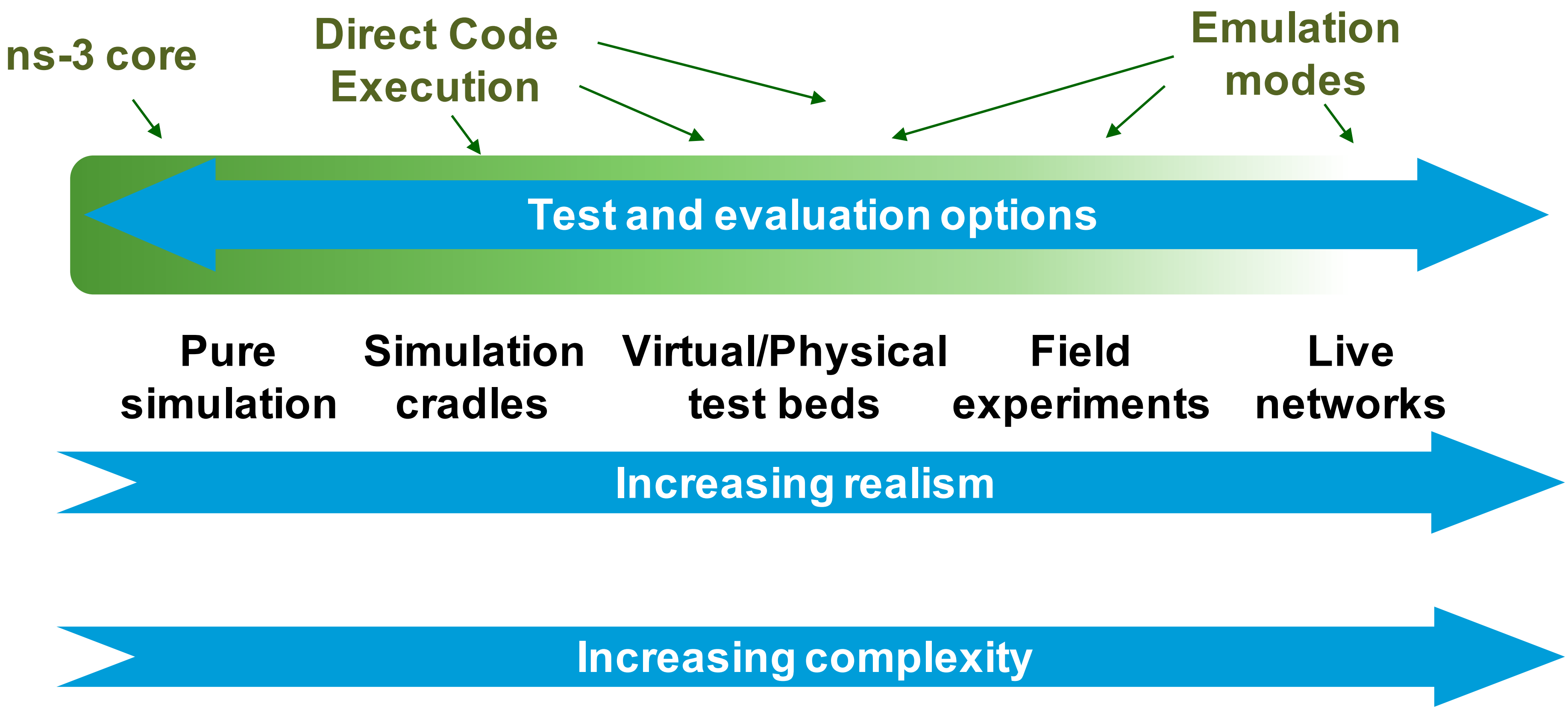
Runs on a single machine



or partitioned across a cluster

Network performance evaluation options

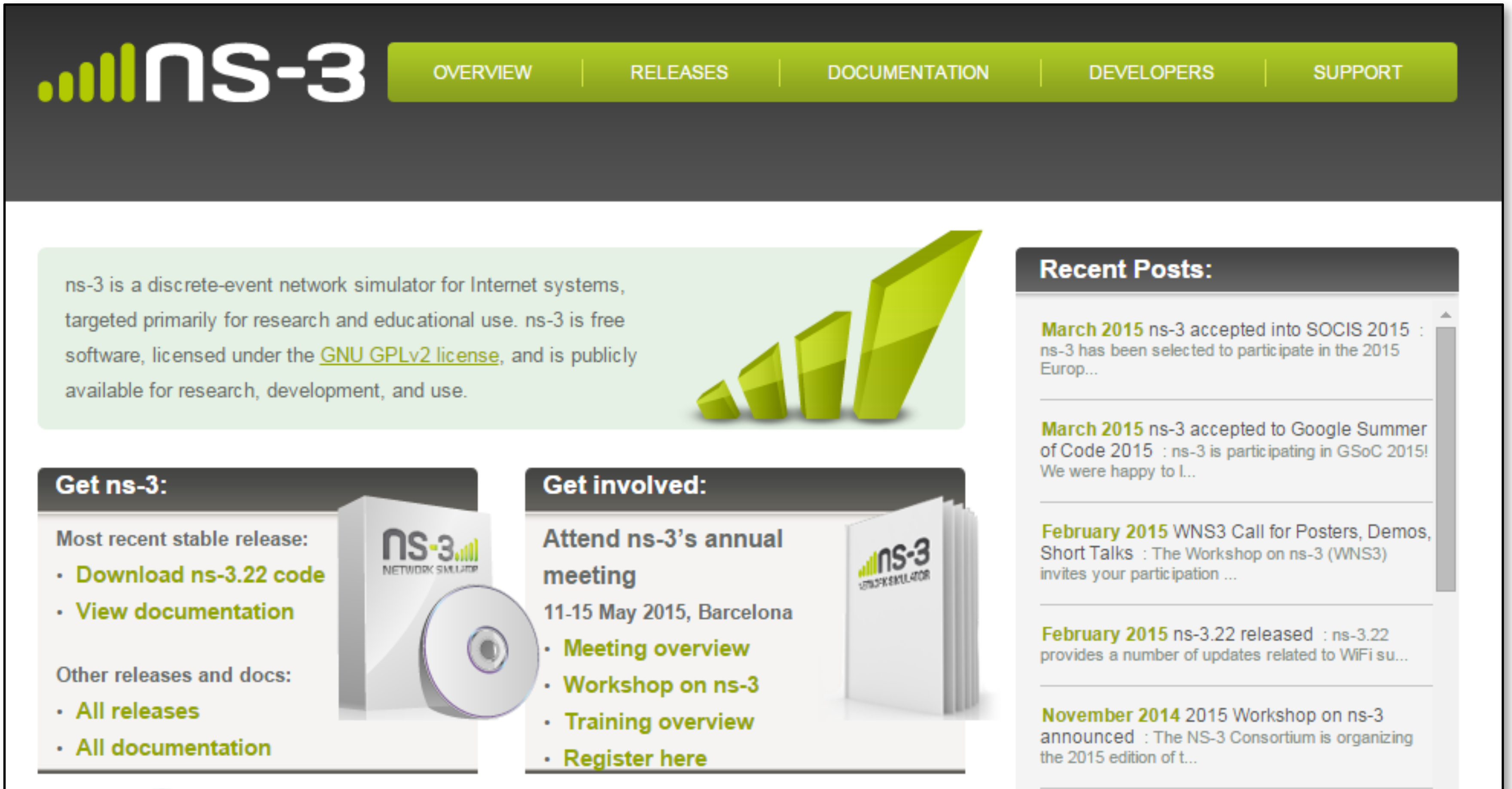
- ns-3 enables researchers to more easily move between simulations, test beds, and experiments



- The open-source project

ns-3 main website

- Project home: https://www.nsnam.org



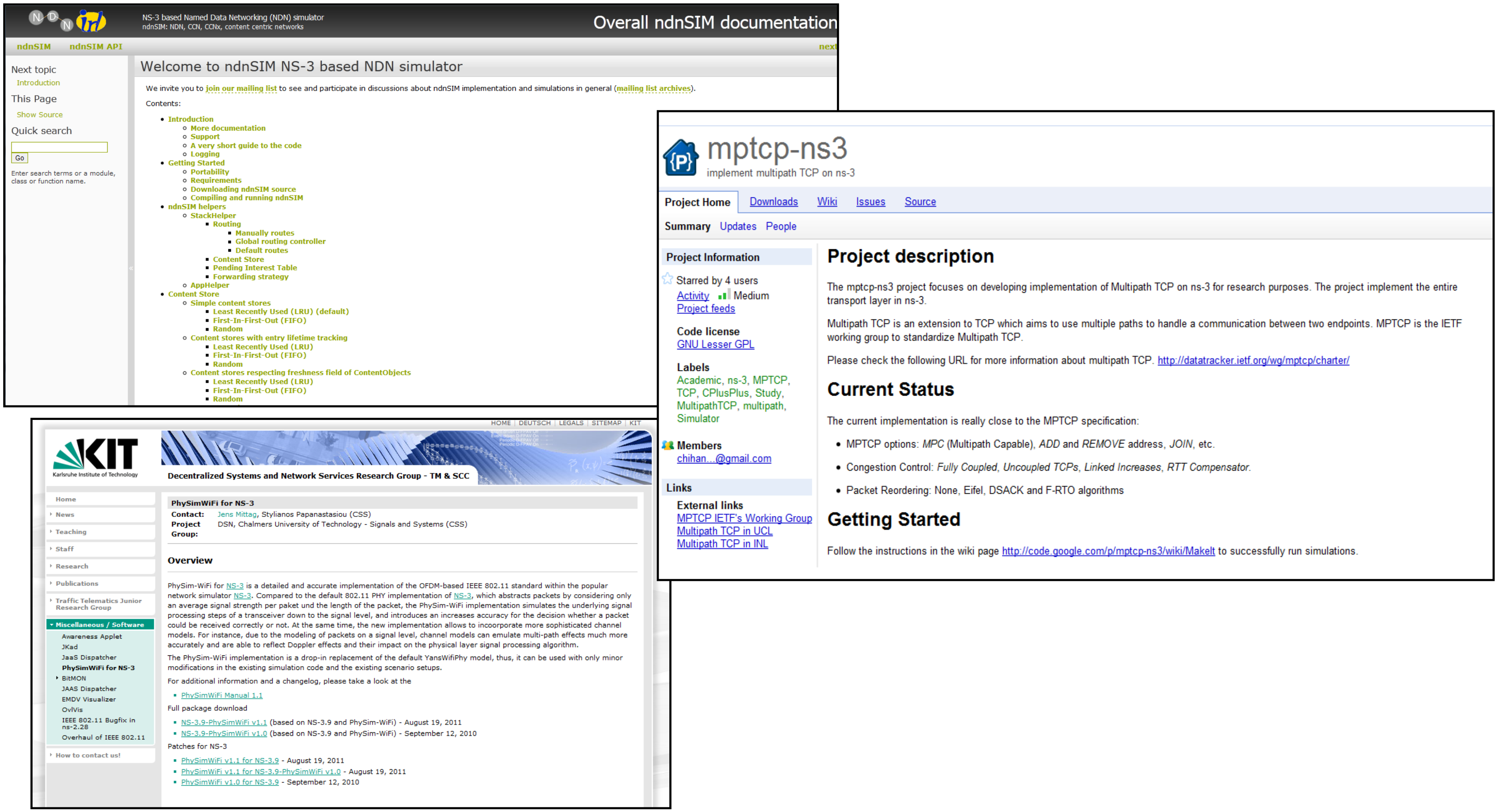
How the project operates

- Project provides three annual software releases
- Users interact on mailing lists and using Bugzilla bug tracker
- Code may be proposed for merge
  - Code reviews occur on a Google site
- Maintainers (one for each module) fix or delegate bugs, participate in reviews
- Project has been conducting annual workshop and developer meeting around SIMUTools through 2013
  - Some additional meetings on ad hoc basis
- Summer projects (Google Summer of Code, ESA Summer of Code in Space, others...)

Maintainers, Authors, Users

- ~10-15 maintainers at any given time
- 191 authors credited in AUTHORS file
- Over 6000 subscribers to ns-3-users Google Groups forum
- Over 1500 subscribers to ns-developers mailing list
- Various project forks exist (on Github and elsewhere)

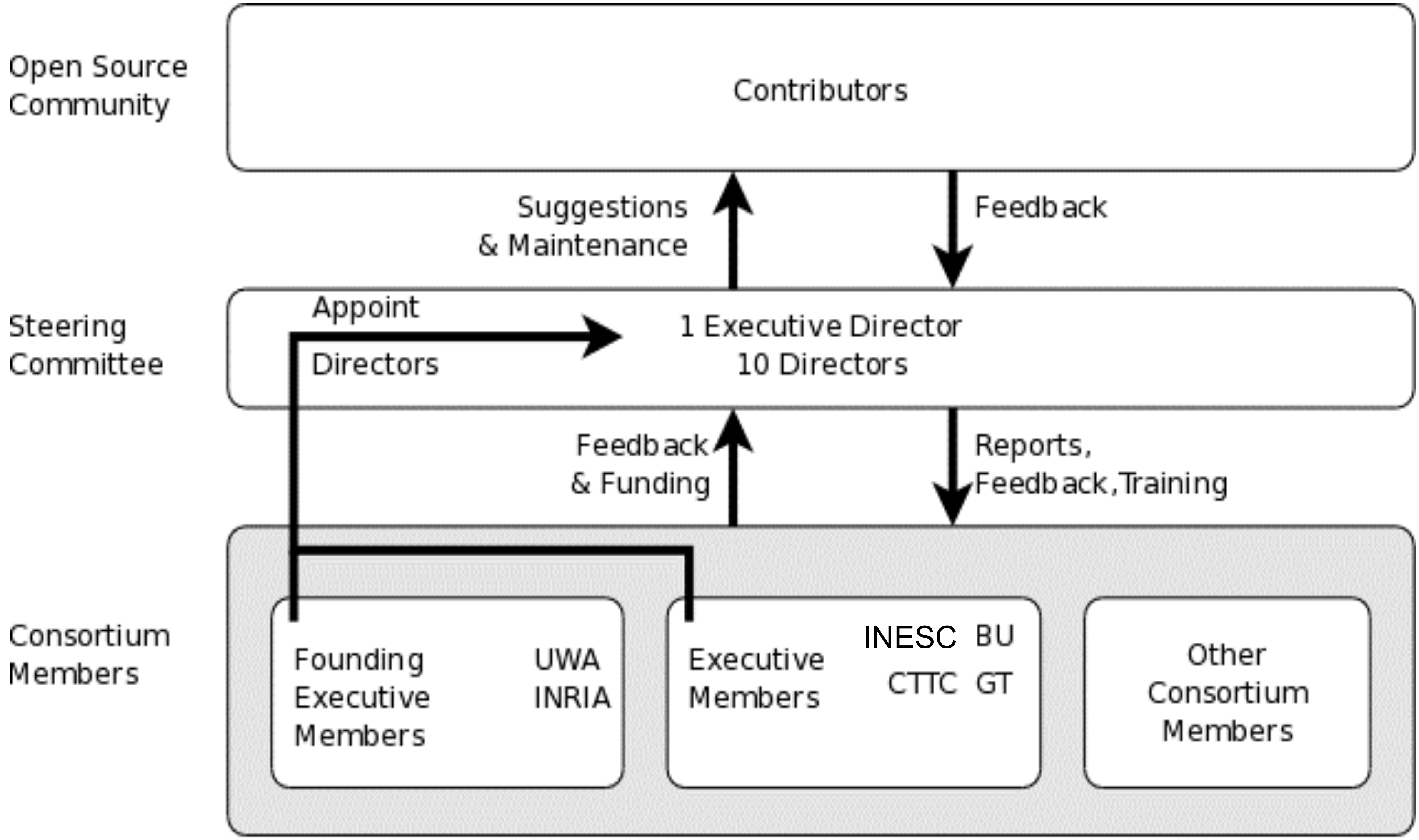
Contributed code and associated projects



Sustainment

- The NS-3 Consortium is a collection of organizations cooperating to support and develop the ns-3 software.
- It operates in support of the open source project
  - by providing a point of contact between industrial members and ns-3 developers,
  - by sponsoring events in support of ns-3 such as users' days and workshops,
  - by guaranteeing maintenance support for ns-3's core, and
  - by supporting administrative activities necessary to conduct a large open source project.

ns-3 Consortium governance



Acknowledgment of support

National Science Foundation
WHERE DISCOVERIES BEGIN

informatics mathematics

PLANETE

Georgia Tech

Google SUMMER OF CODE 2012

INESCPORTO®
LABORATÓRIO ASSOCIADO

CTTC
Centre
Tecnològic
de Telecomunicacions
de Catalunya

esa

ISI
Information Sciences Institute

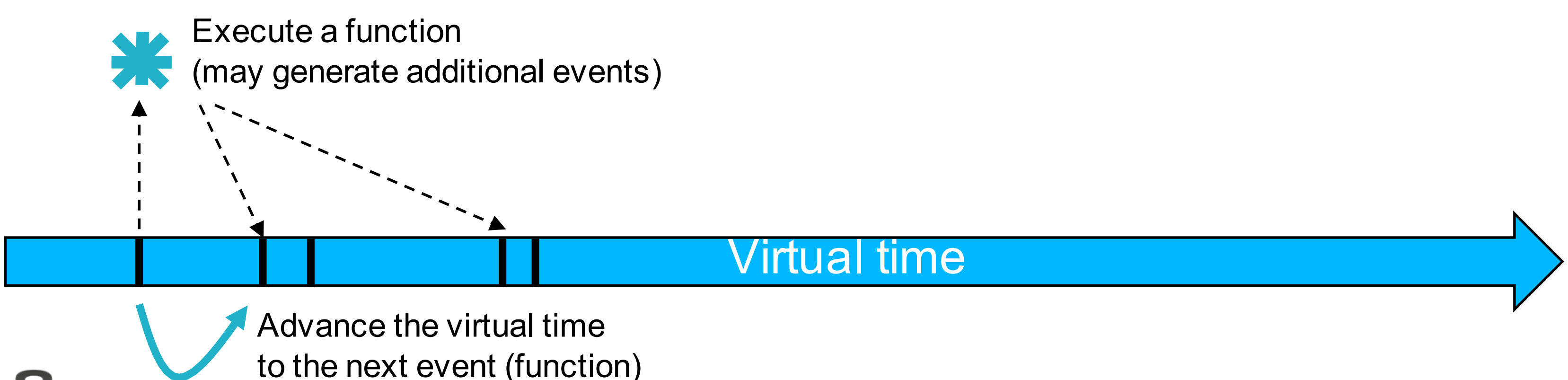
- Software overview

Software overview

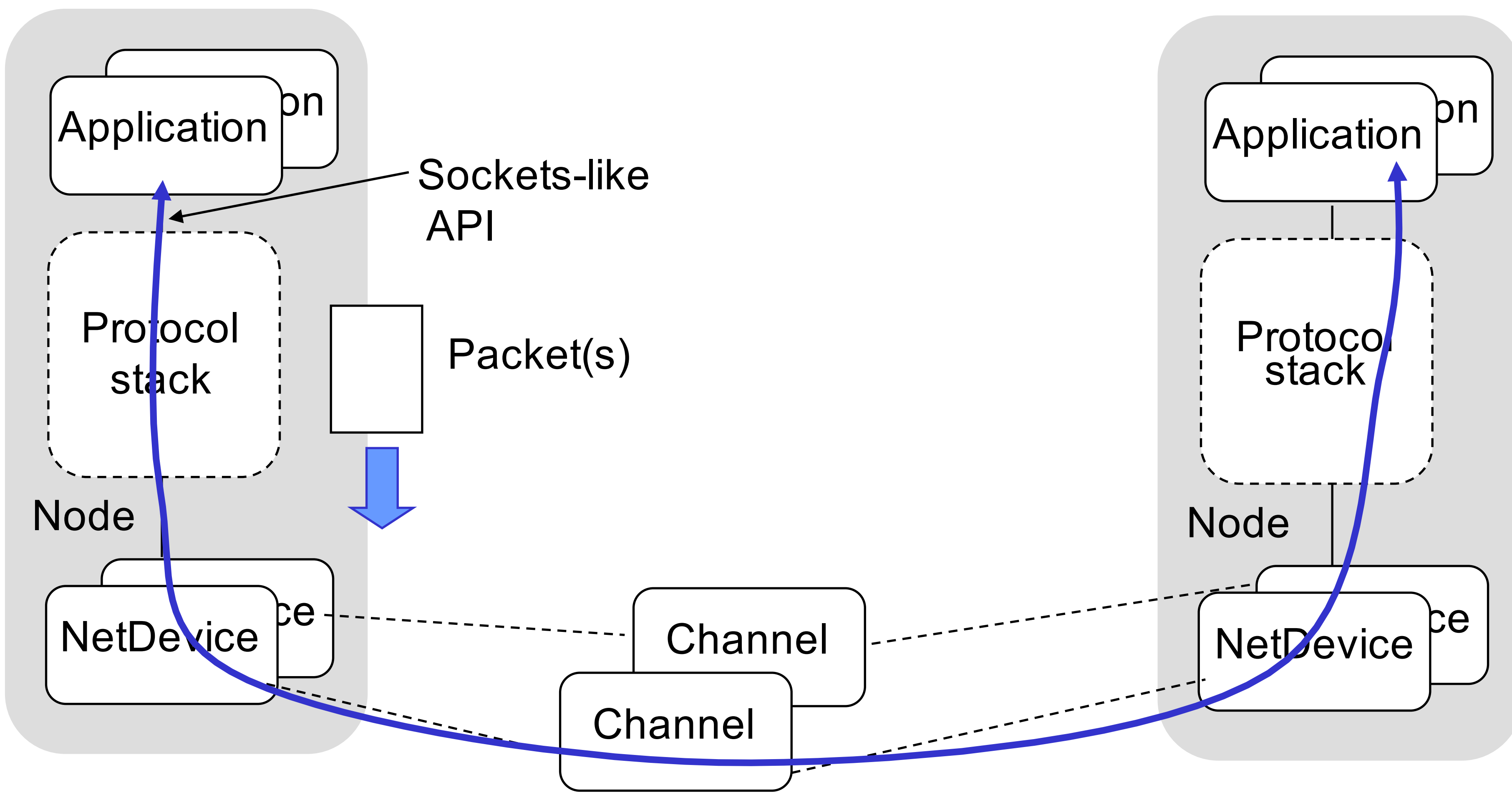
- ns-3 is written in C++, with bindings available for Python
  - simulation programs are C++ executables or Python programs
  - ~350,000 lines of C++ (cloc estimate)
  - almost exclusively C++98, beginning to use C++11
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is mainly supported for Linux, OS X, and FreeBSD
  - Windows Visual Studio port available
- ns-3 is not backwards-compatible with ns-2

Discrete-event simulation basics

- Simulation time moves in discrete jumps from event to event
- C++ functions schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- Simulation::Run() executes a single-threaded event list
- Simulation stops at specific time or when events end



The basic ns-3 architecture



Software orientation

Key differences from other network simulators:

1) Command-line, Unix orientation
   - vs. Integrated Development Environment (IDE)

2) Simulations and models written directly in C++ and Python
   - vs. a domain-specific simulation language

ns-3 does not have a graphical IDE

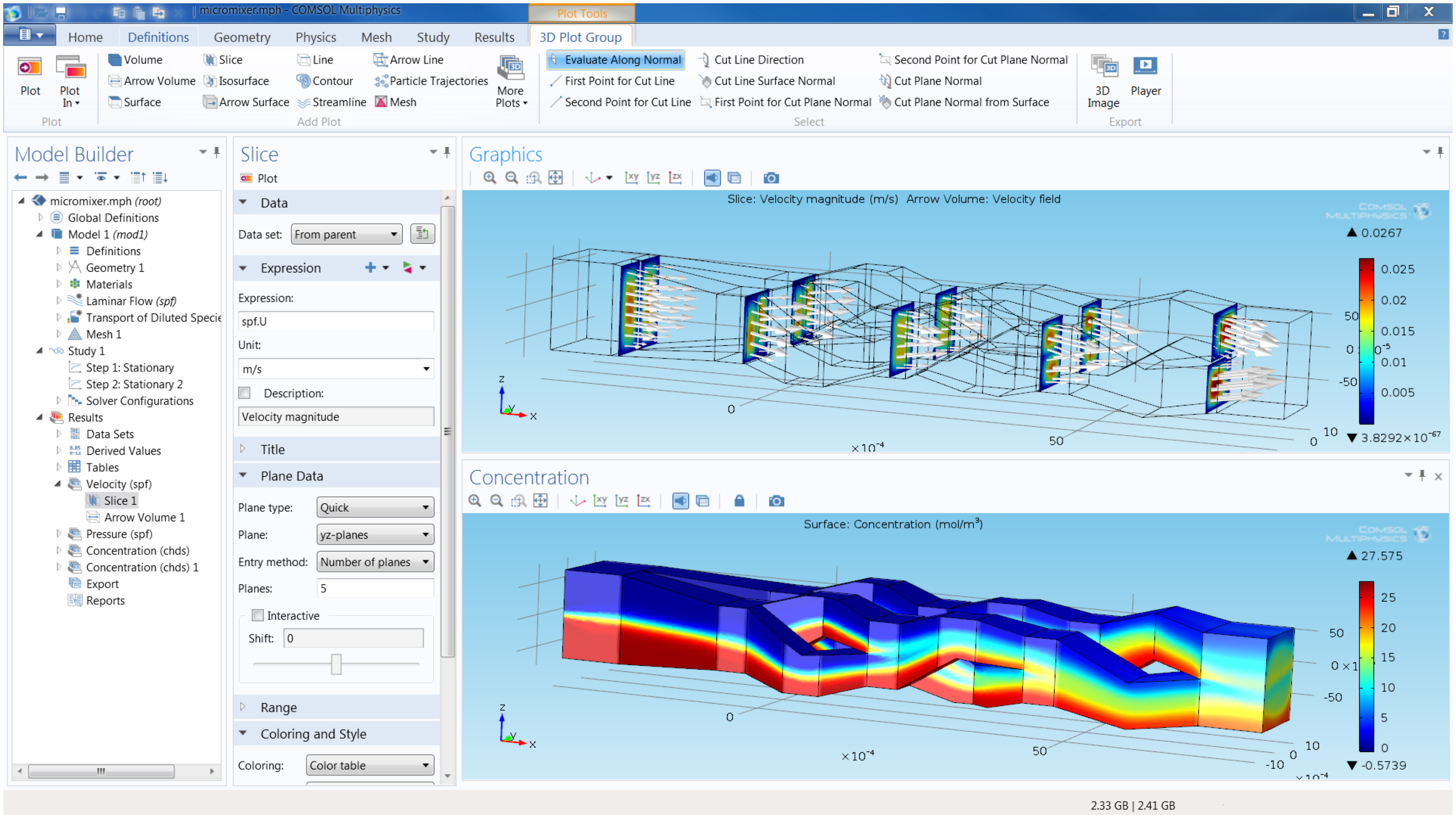


Figure source: https://www.comsol.com/comsol-multiphysics

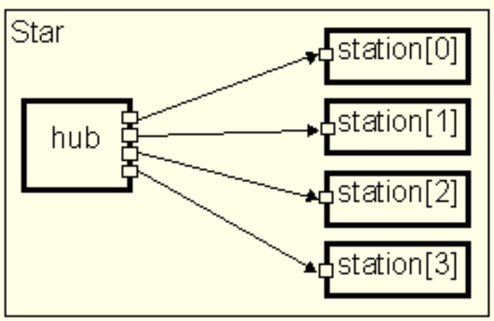
ns-3
NETWORK SIMULATOR

ns-3 not written in a high-level language

Submodule vectors, gate vectors and multiple connections are illustrated in the following example:

```
simple Hub
  gates:
    out: outport[];
endsimple

simple Station //...

module Star
  submodules:
    hub: Hub
      gatesizes: outport[4];
    station: Station[4];
  connections:
    for i=0..3 do
          hub.outport[i] --> station[i].in;
    endfor
endmodule
```

The result of the above is depicted in Fig.4.

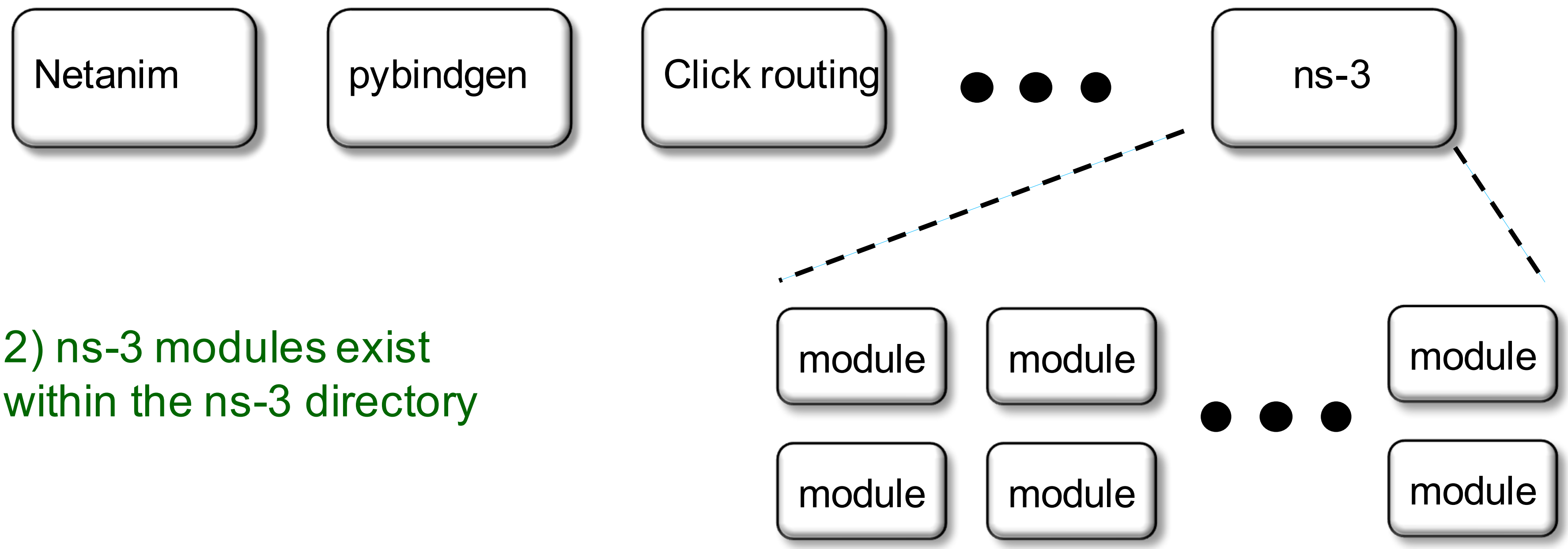


Example of OMNeT++ Network Description (NED) language
Figure excerpted from http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm

Software organization

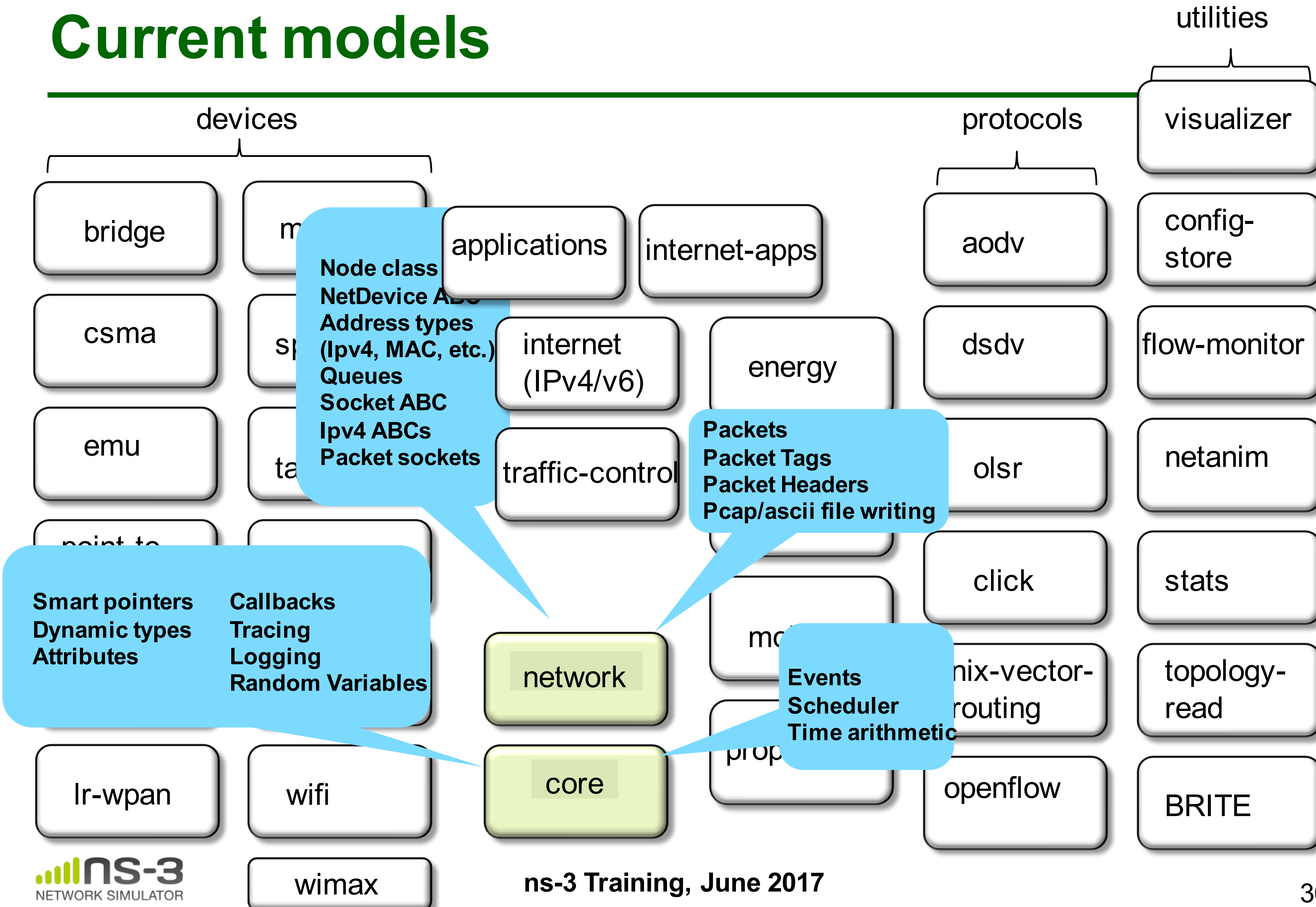
- Two levels of ns-3 software and libraries

1) Several supporting libraries, not system-installed, can be in parallel to ns-3



2) ns-3 modules exist within the ns-3 directory

Current models



Module organization

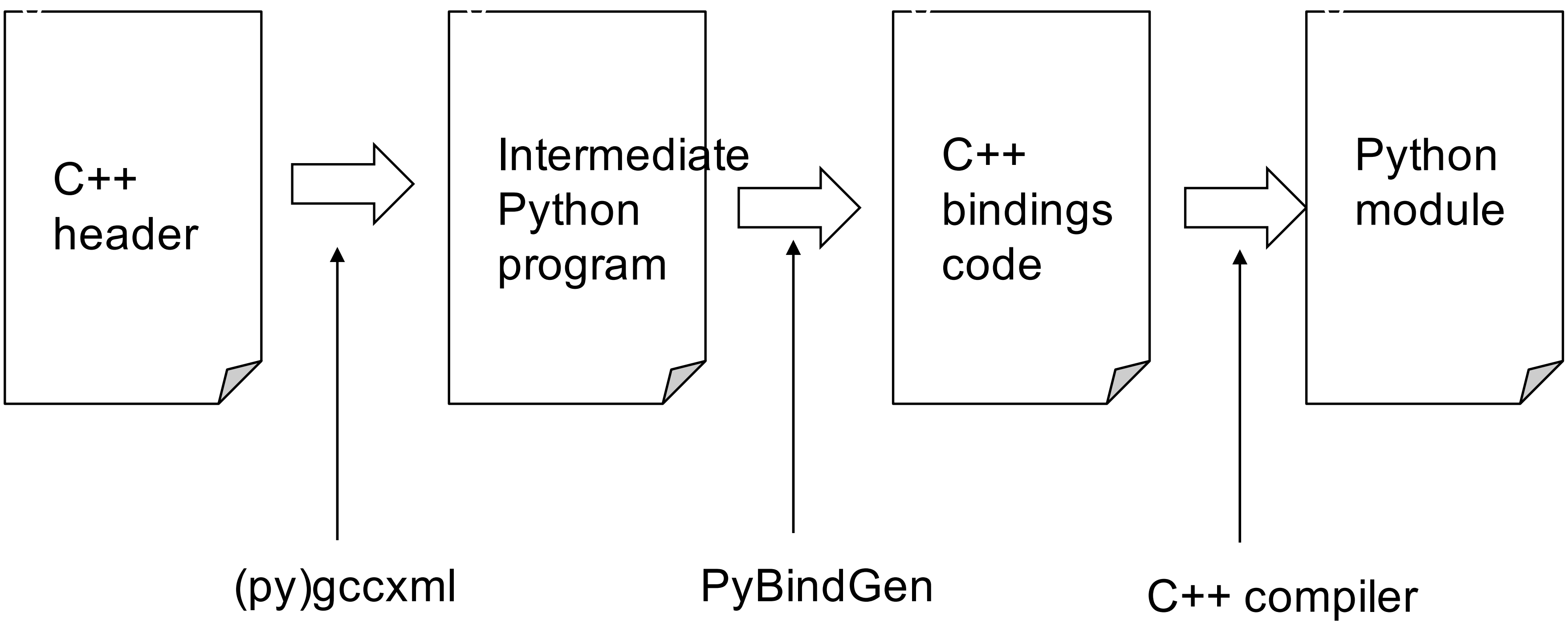
- models/
- examples/
- tests/
- bindings/
- doc/
- wscript

ns-3 programs

- ns-3 programs are C++ executables that link the needed shared libraries
  - or Python programs that import the needed modules
- The ns-3 build tool, called 'waf', can be used to run programs
- waf will place headers, object files, libraries, and executables in a 'build' directory

Python bindings

- ns-3 uses a program called PyBindGen to generate Python bindings for all libraries



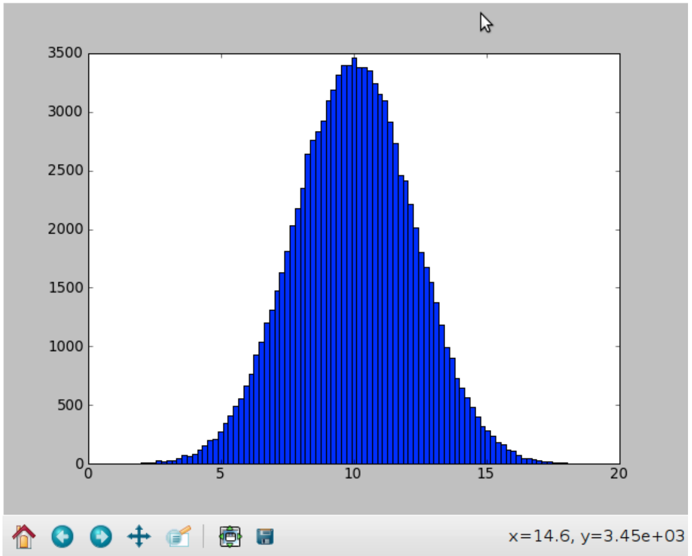
Integrating other tools and libraries

Other libraries

- more sophisticated scenarios and models typically leverage other libraries
- ns-3 main distribution uses optional libraries (libxml2, gsl, mysql) but care is taken to avoid strict build dependencies
- the 'bake' tool (described later) helps to manage library dependencies
- users are free to write their own Makefiles or wscripts to do something special

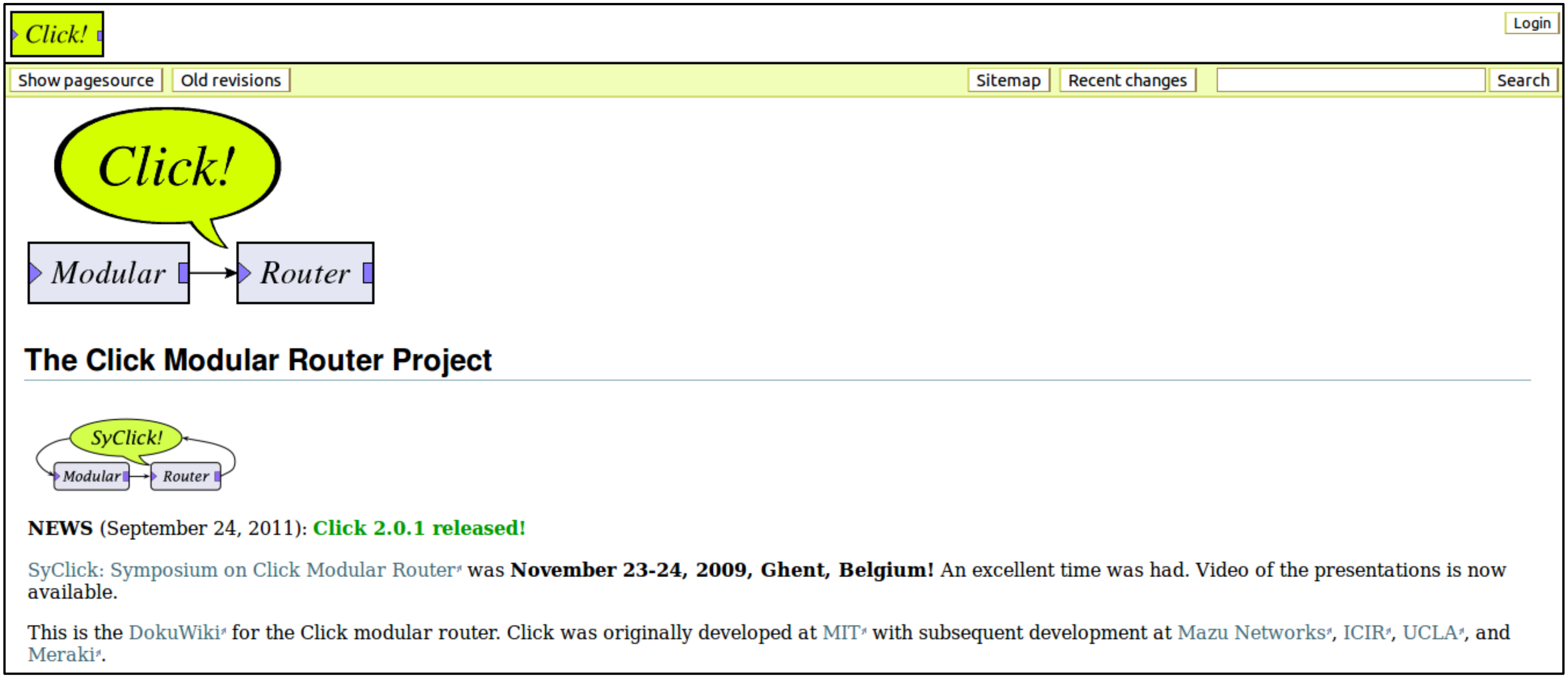
Matplotlib

- `src/core/examples/sample-rng-plot.py`




```
# Demonstrate use of ns-3 as a random number generator integrated
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial

import numpy as np
import matplotlib.pyplot as plt
import ns.core

# mu, var = 100, 225
rng = ns.core.NormalVariable(100.0, 225.0)
x = [rng.GetValue() for t in range(10000)]

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)

plt.title('ns-3 histogram')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Click Modular Router



Click!

Login

Show pagesource | Old revisions | Sitemap | Recent changes | Search

Click!

Modular → Router

The Click Modular Router Project

SyClick!

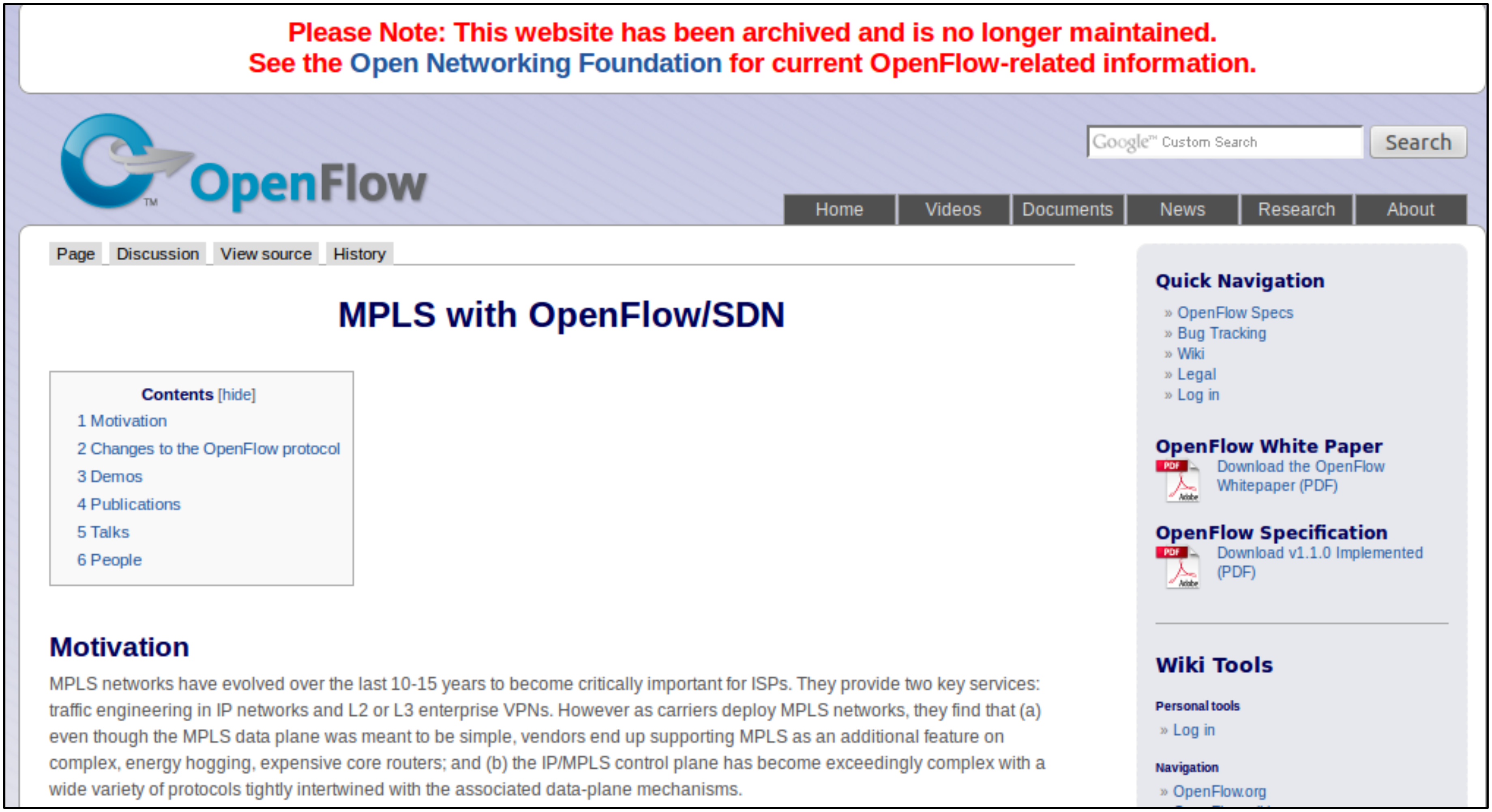
Modular → Router

NEWS (September 24, 2011): **Click 2.0.1 released!**

SyClick: Symposium on Click Modular Router was **November 23-24, 2009, Ghent, Belgium!** An excellent time was had. Video of the presentations is now available.

This is the DokuWiki for the Click modular router. Click was originally developed at MIT with subsequent development at Mazu Networks, ICIR, UCLA, and Meraki.

OpenFlow Switch



Please Note: This website has been archived and is no longer maintained.
See the Open Networking Foundation for current OpenFlow-related information.

OpenFlow™

Search

Home | Videos | Documents | News | Research | About

Page | Discussion | View source | History

MPLS with OpenFlow/SDN

Contents [hide]

Motivation

MPLS networks have evolved over the last 10-15 years to become critically important for ISPs. They provide two key services: traffic engineering in IP networks and L2 or L3 enterprise VPNs. However as carriers deploy MPLS networks, they find that (a) even though the MPLS data plane was meant to be simple, vendors end up supporting MPLS as an additional feature on complex, energy hogging, expensive core routers; and (b) the IP/MPLS control plane has become exceedingly complex with a wide variety of protocols tightly intertwined with the associated data-plane mechanisms.

Quick Navigation

- » OpenFlow Specs
- » Bug Tracking
- » Wiki
- » Legal
- » Log in

OpenFlow White Paper

Download the OpenFlow Whitepaper (PDF)

OpenFlow Specification

Download v1.1.0 Implemented (PDF)

Wiki Tools

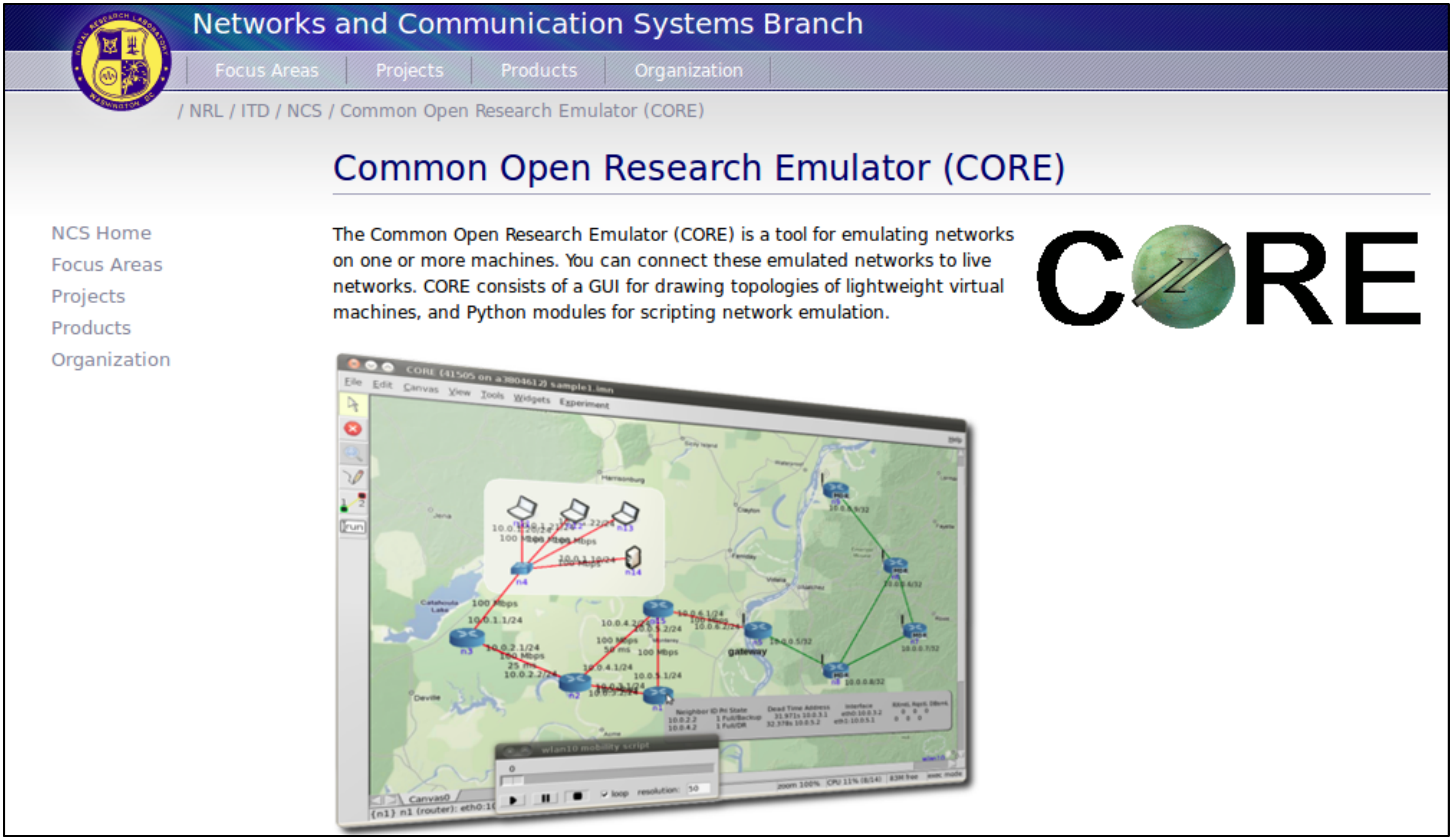
Personal tools

- » Log in

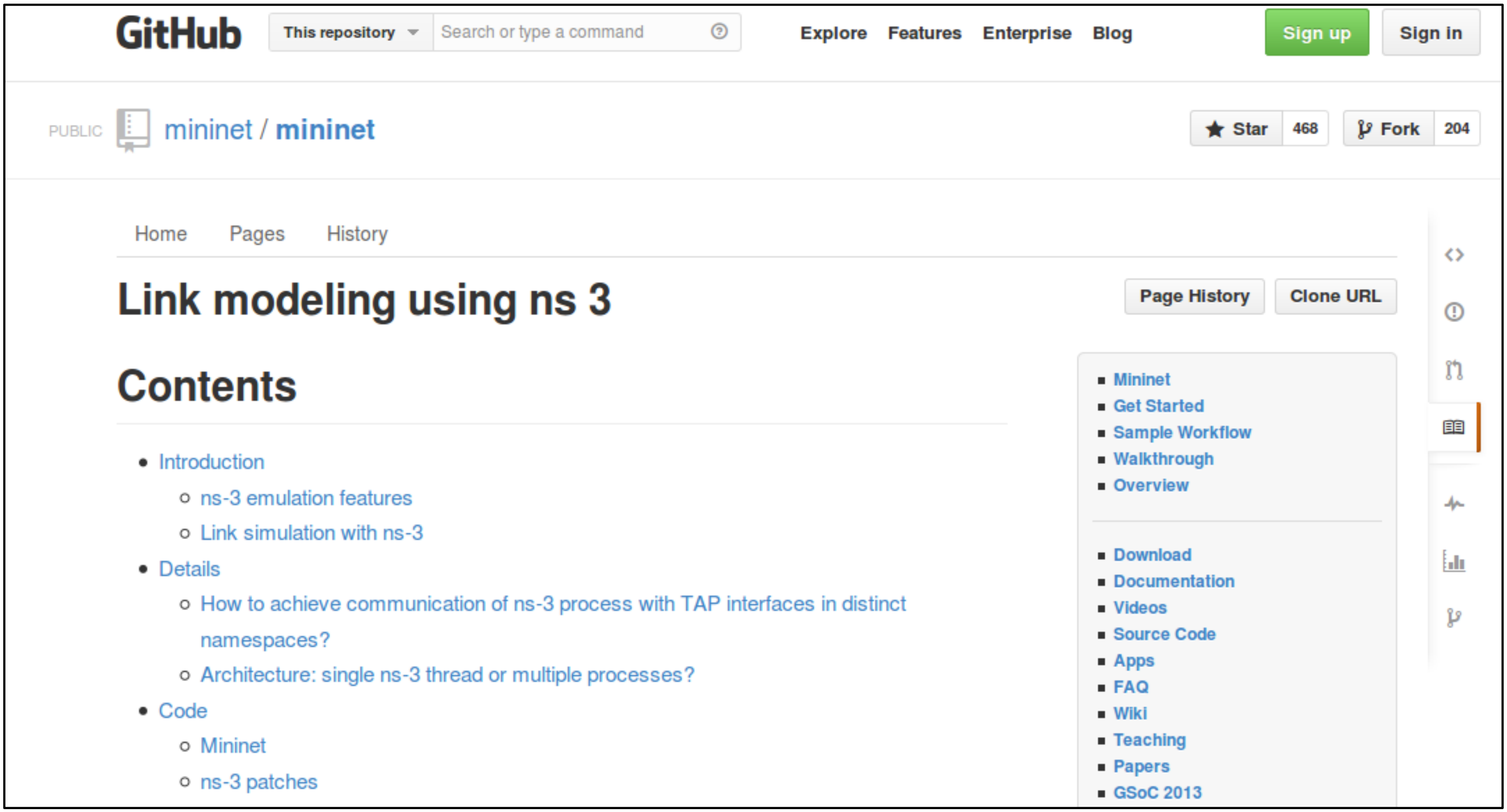
Navigation

- » OpenFlow.org

CORE emulator



mininet emulator



Co-simulation frameworks have emerged

- PNNL's FNCS framework integrates ns-3 with transmission and distribution simulators

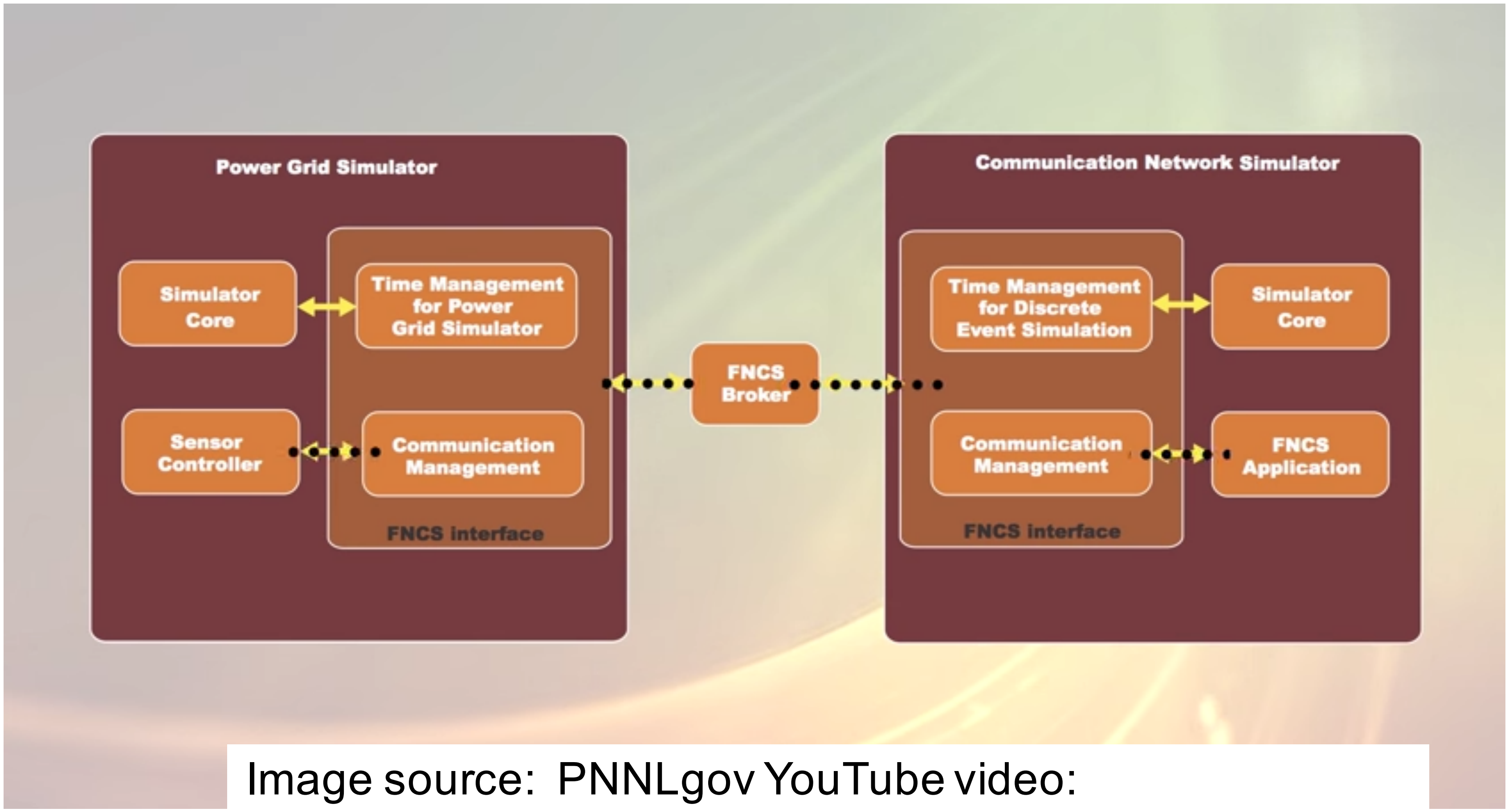


Image source: PNNLgov YouTube video:
Introducing FNCS: Framework for Network Co-Simulation

FAQs

- Does ns-3 have a Windows version?
  - Yes, for Visual Studio 2012
  - http://www.nsnam.org/wiki/Ns-3_on_Visual_Studio_2012
- Does ns-3 support Eclipse or other IDEs?
  - Instructions have been contributed by users
  - http://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3
- Is ns-3 provided in Linux or OS X package systems (e.g. Debian packages)?
  - Not officially, but some package maintainers exist
- Does ns-3 support NRL protolib applications?
  - Not yet

Summarizing

- ns-3 models are written in C++ and compiled into libraries
  - Python bindings are optionally created
- ns-3 programs are C++ executables or Python programs that call the ns-3 public API and can call other libraries
- ns-3 is oriented towards the command-line
- ns-3 uses no domain specific language
- ns-3 is not compatible with ns-2